Extending UML State Diagrams to Model Agent Mobility

Feras Ahmad Hanandeh[1] and Majdi Yousef Al-Shannag[2]
[1]Department of Computer Information Systems, Faculty of Prince Al-Hussein Bin Abdallah II For Information Technology, Hashemite University, Jordan.
Email:Feras@hu.edu.jo
[2]Department of Computer Information Science, Faculty of Information Technology
Yarmou University, Jordan.
Email:majdis@yu.edu.jo

***Abstract*— This paper presents a simplified form of UML state diagrams for modeling agent mobility. Mobile agent has gained more importance technology. The notations used to model agent mobility are focused on capturing agent creation, mobility paths and current agent location. In this paper, we demonstrate how the simplification of the state UML 2.0 Activity Diagrams can be used for modeling mobile agent applications. The paper concludes with the appropriateness of the presented approach for modeling agent mobility with UML state diagrams as well as with sequence diagrams of the mobile agent system.**

***Index Terms*— UML State Diagrams, Agent Mobility, UML Sequence Diagrams**

I. Introduction

This research extends UML 2.0 states diagrams to the one of the most important new concepts which is the mobile computing which gains more and more interest. The objective behind this research is that agent concepts and mobile software agents have become part of the system and service architecture of new generation networks, systems and services. The main goal of the presented approach is to successfully model such a dynamic environment with a good representation of agent mobility and execution paths. The advent of network technology has prompted new computing and communication paradigm in which codes and processes can move over the network. Mobile agents are autonomous computing entities (codes/processes), which can autonomously decide where to move over the network and what tasks to perform at different hosts [3]. A large number of programming languages, systems and APIs have emerged to support this paradigm; so there has been an emerging trend to use Unified Modeling Language (UML) for modeling agent-based systems in general. The most notable work in this arena is Agent UML (AUML) [7] in which UML is extensively used to model all aspects of agent. Agent technology enables the realization of complex software systems characterized by situation awareness and intelligent behavior, and can open the way to new application domains while supporting the integration of existing and new software, and make the development process for such applications easier and more flexible. However, deploying agent technology successfully in industrial applications requires industrial-quality software methods and explicit engineering tools, such as UML 2.0. The UML is gaining wide acceptance for the representation of engineering artifacts in object-oriented software. Our view of agents as the next step beyond objects leads us to explore extensions to UML and idioms within UML to accommodate the distinctive requirements of agents. The result is an AUML [5]. Agent interaction protocols are used to model the interaction between agents. It is important to model the complex logic, including data flow, within a software agent [6]. To overcome these limitations, we propose using the UML 2.0 [8] activity diagram to specify action plans. This diagram models the system behavior by the state diagram. Actions are considered the basic units of the system behavior. The activity diagram is the most noticeable change in UML 2.0 [6]. The paper is organized as follows: Section 2 presents related work. Section 3 elaborates modeling agent mobility using UML sequence diagrams. In section 4 an approach for modeling agent mobility using state diagrams is presented. Section 5 concludes the paper.

II. Related Work

Mario Kusek and Gordan Jezic in [1] presented a proposal for modeling agent mobility with UML sequence diagrams. They described and compared four approaches according to their clarity, the space needed for graphics and their expression of mobility. The stereotyped mobility diagram gives an overall view of nodes and agent mobility paths. The swimlaned mobility diagram gives a clear representation of agent mobility, current locations and creation. Both stereotyped and swimlaned mobility diagrams clearly represent agent execution and mobility paths, but they are not suitable for modeling the systems with a large number of nodes and agents. In state representation mobility diagrams, mobility is represented by changing the state of a moving agent. In frame fragment mobility diagrams, each frame fragment of a sequence diagram represents the execution at a node. The state representation and frame fragment mobility diagrams have poorer representations of migration and execution paths. These diagrams are suitable for modeling multi–agent systems with mobile agents and a large number of nodes. An advantage of these approaches is a better overall view of agent roaming and current agent location, but in some cases it is not possible to order agents in such a way that one frame fragment can represent all the agents at a certain node [1]. Hubert Baumeister and et al. in [2] presented an extension to UML

DOI: 01.IJCOM.02.03. 521

class and activity diagrams to model mobile systems and they assumed that mobile objects can migrate from one location to another. Locations can be nested and mobile too. They introduced stereotypes to model mobile objects, locations, and activities like moving or cloning, and they introduced two notational variants of activity diagrams for modeling mobility. They found that locations, mobile objects, and the atLoc relation could be modeled explicitly as abstract classes and associations in class diagrams and mobile objects, like Passenger and Planes, would inherit from these classes [2]. Miao Kang and et al. in [3] proposed an extension of activity diagrams in UML 1.5 for modeling mobile agent applications and found that the latest version of UML 2.0, which provides better model elements in activity diagrams, is more effective and flexible in modeling mobile agent applications. This observation prompted them to upgrade their work in UML 1.5 to UML 2.0, and they demonstrated how UML 2.0 activity diagrams can be used for modeling mobile agent applications and discussed their underlying computational models which capture mobility of agents. They used multidimensional hierarchical partitions to model locations and agents respectively, which help to visualize the algorithmic behavior of multi agent systems with locality [3]. Bernhard Bauer and James Odell in [4] shown how UML 2.0 can be applied for the specification of agent-based systems, and they given a short overview on existing agent methodologies to have a reference what has to be specified in such systems. They found that many of the errors and inconsistencies of the original submission have been rectified, and more than 3000 issues were files and resolved by the UML 2.0 Finalization Task Force. As such software vendors can begin to build software tools that support the UML 2.0 Superstructure and Infrastructure [4]. James Odell and et al. in [5] addressed two requirements (relating the use of agents to the nearest antecedent technology "object-oriented software development" and using artifacts to support the development environment throughout the full system lifecycle) by describing some of the most common requirements for modeling agents and agent-based systems using a set of UML idioms and extensions. They illustrated the approach by presenting a three-layer AUML representation for agent interaction protocols and conclude by including other useful agent-based extensions to UML. They suggested several straightforward UML extensions that support the additional functionality that agents offer over the current UML version 1.3. Many of these proposed extensions are already being considered by the OO community as useful extensions to OO development on UML version 2.0 [5]. Viviane Torres da Silva and et al. in [6] proposed the use of UML 2.0 activity diagrams to model agent plans and actions. They considered a plan to be an activity. Both plans and activities are composed of actions and define the action execution sequence. They demonstrated how these diagrams can be applied to model agent plans and actions by using some features available in the UML 2.0 activity diagrams and defining some new ones. They seen that it is possible to describe actions using a domain-independent notation, to associate goals and roles with plans, to check the information in the agent mental state by using guard conditions, to describe messages, to represent agents changing their roles, to describe the actions that are duties and rights, to model agents moving from an organization to another and to model agents moving from an environment to another [6]. Dianxiang Xu and Yi Deng in [9] proposed a two-layer approach for the formal modeling of Logical Agent Mobility (LAM) using predicate/transition (PrT) nets. They viewed a mobile agent system as a set of agent spaces and agents could migrate from one space to another. Each agent space is explicitly abstracted to be a component, consisting of an environmental part and an internal connector dynamically binding agents with their environment. They used a system net, agent nets, and a connector net to model the environment, agents, and the connector, respectively. They presented a case study of modeling and analyzing an information retrieval system with mobile agents. They obtained that the modeling of migration process has led to some insights into logical code mobility of software agents as well as the differences from physical mobility in an ad hoc networks. In terms of migration request, two styles of migration, autonomous and passive, are formally identified. Strong mobility is made explicit by ensuring state preservation during migration. Agents are disabled when deactivated upon migration and disconnected from environment. Migration as atomic change of location attribute is inadequate for logical code mobility, though location is a critical concept. Managing locations by environments, rather than by agents themselves, facilitates the investigation of agent transfer [9]. Jacques Ferber and Olivier Gutknecht in [10] presented a generic meta-model of multi-agent systems based on organizational concepts such as groups, roles and structures, which called AALAADIN, that defines a very simple description of coordination and negotiation schemes through multi-agent systems. This meta-model allows for agent heterogeneity in languages, applications and architectures. They introduce the concept of organizational reflection which uses the same conceptual model to describe system level tasks such as remote communication and migration of agents. AALAADIN is able to overcome these problems by allowing designers of multi agent systems to describe any kind of organization using only the core concepts of groups, agents and roles. They also presented a development platform, called MadKit, in which all these concepts have been implemented [10]. Marcus Huber in [11] presented a hybrid intelligent agent architecture (JAM) that draws upon the theories and ideas of the Procedural Reasoning System (PRS), Structured Circuit Semantics (SCS), and Act plan interlhtgua. JAM draws upon the implementation pragmatics of the University of Michigan's and SRI International's implementation of PRS (UMPRS and PRS-CL, respectively). The JAM agent architecture also provides an agentGo primitive function utilizing Java's object serialization mechanism to provide widely-supported mobility capabilities. Huber [11] extended the JAM plan representation to include declarative representations for individual primitive actions and implementing partial order planning algorithms based upon the new representations.

DOI: 01.IJCOM.02.03.521

III. Modeling Agent Mobility with Sequence Diagrams

A sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

A. The Sequence Diagram of the UML 2 in Model Agent Mobility

- Stereotyped Mobility Diagram:

The stereotyped mobility diagram [1] introduced three stereotypes (Agent <<agent>>, Location<<at>> and Move <<move>>) as presented in Figure 1. The diagram initiated by locating the agent at node n1 which is indicated with a message with stereotype <<at>>. After that, agent moves from this location to location at node n2, and that is represented with message and stereotype <<move>>. When the agent creates a new agent, it is indirectly done by sending message to node where the new agent should be created.

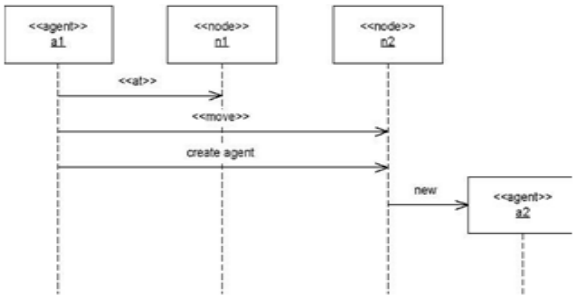

Figure 1.The Stereotyped Mobility Diagram representing 2 Agents and 2 Locations

B. Case study: Simple price searcher diagram

This diagram as presented in [1] represents three network nodes: Home, Host1 and Host2 (store agent), which is responsible for providing pricelist. The Searcher agent is *created* at the Home node. The Searcher agent *move* from Home node to Host1 node and requests Store1 agent to be provided by the price list. The Store1 agent responds with the whole pricelist. The Searcher extracts the price for the item and *move* to the next node. After visiting all nodes the Searcher agent *move* back to the Home node and informs the user where and how much the price of the specified item.

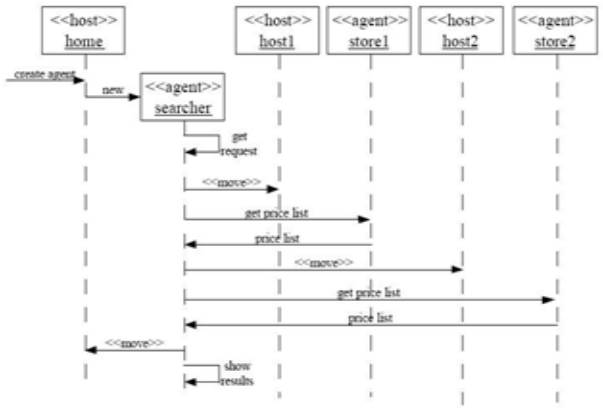

Figure 2. The Stereotyped Mobility Diagram for the Scenario

IV. Modeling Agent Mobility with State Diagrams

A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams are used to give an abstract description of the behavior of a system. This behavior is analyzed and represented in series of events that could occur in one or more possible states. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.

A. The State Diagram of the UML 2 in Model Agent Mobility

Representing agent mobility by state diagram is a complicated process when there are many agents with many locations. Figure 3 represents agent mobility state diagram for 2 agents and 2 locations.

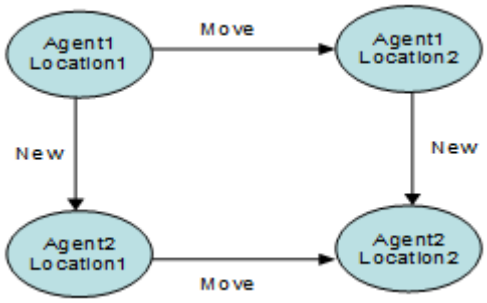

Figure 3.Agent Mobility State Diagram for 2 Agents and 2 Locations

The diagram shows that Agent1 move from Loaction1 to Location2 (change its state) by the action Move and creates a new agent (Agent2) by the action New. Adding a new location (Location3) means that extracting the previous diagram to the diagram in Fig. 4:

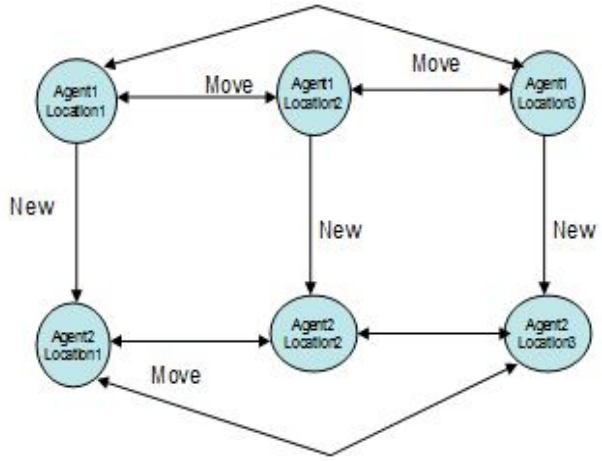

Figure 4.Agent Mobility State Diagram for 2 Agents and 3 Locations

Again adding a new agent (Agent 3) means that extracting the previous diagram to diagram in Fig. 5:

DOI: 01.IJCOM.02.03.521

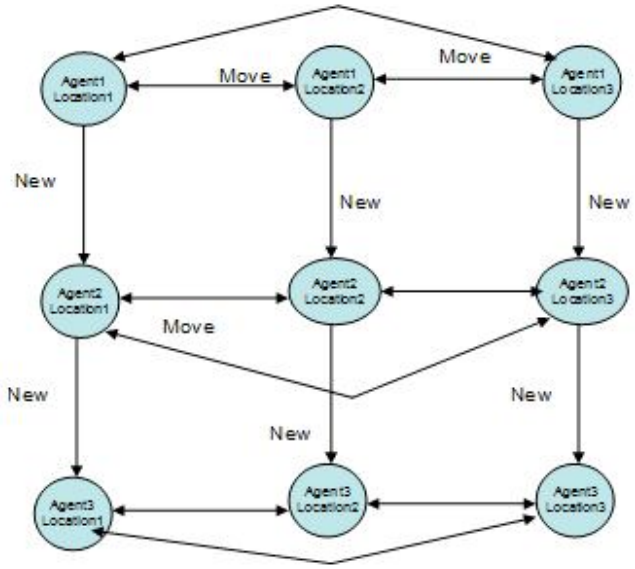

Figure 5.Agent Mobility State Diagram for 3 Agents and 3 Locations

As the number of agent's and locations increases, the agent mobility state diagram becomes huge and reading is difficult because of many relating parameters. Within such an environment, agents repeatedly interact with one another, and each agent's actions affect the joint state of all agents, which in turn make the agent mobility state diagram complex.

B. Simplified State Diagram of the UML 2 in Model Agent Mobility

As a solution of the above state diagram drawbacks, the suggested approach of this article use two arrays, one for Agents, A[i] and the other for Locations, L[j]. Along with, adding a new computation notation to increase the index of the agents array and/or of the locations array as presented in Figure 6.

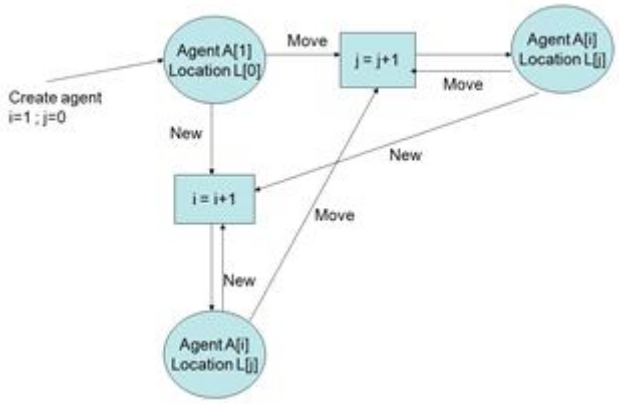

Figure 6.Agent Mobility State Diagram for n Agents and m Locations

The diagram initiated by locating the created agent; A[1] at location; L[0]. After that, once the agent moves from this location to another location by the action Move, j is increased by one and the agent will be located at location L[j+1]. When the agent creates a new agent, it is indirectly done by the action New which increases i by one such that the new agent is A[i+1].

V. Conclusion

As the mobile agent system is an important technology, we demonstrate how state UML 2.0 Activity Diagrams can be used for modeling it. We use notations for the model agent mobility to capturing agent creation, mobility paths and current agent location. The presented approach extracts the State Diagram that can be used for multi locations and agents. The simplified state diagram presented in this research competes with the sequence diagrams for modeling mobile agent applications.

References

[1] M. Kusek and G. Jezic. "Extending UML Sequence Diagrams to Model Agent Mobility", AOSE'06 Proceedings of the 7th international conference on Agent-oriented software engineering VII, 2007.

[2] H. Baumeister, N. Koch, P. Kosiuczenko and M. Wirsing. "Extending Activity Diagrams to Model Mobile Systems", In Proceedings of NODe '02 Revised Papers from the International Conference NetObjectDays on Objects, Components, Architectures, Services, and Applications for a Networked World, 2003.

[3] M. Kang, L. Wang and K. Taguchi. "Modelling Mobile Agent Applications in UML2.0 Activity Diagrams", Proceedings of the 3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems, IEE, Edinburgh, United Kingdom, 2004, pp 104 - 111.

[4] B. Bauer and J. Odell. "UML 2.0 and agents: how to build agent-based systems with the new UML standard", Engineering Applications of Artificial Intelligence 18, 2005, 141–157.

[5] J. Odell, H. Parunak and B. Bauer. "Extending UML for Agents", Proc. of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence, 2000.

[6] V. da Silva, R. Noya and C. de Lucena. "Using the UML 2.0 Activity Diagram to Model Agent Plans and Actions", AAMAS '05 Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, 2005.

[7] B. Bauer, J. P. Muller, and J. Odell. "Agent UML: A Formalism for Specifying Multi agent Software Systems", In Proceedings of the First International Workshop on Agent-Oriented Software Engineering AOSE'00, Limerick, Ireland, LNCS 1957 Springer, 2001, pp. 91-103.

[8] UML: Unified Modeling Language Specification, version 2.0, OMG. Available at: <http://www.uml.org>. Accessed in: May 10, 2011.

DOI: 01.IJCOM.02.03.521